Maple

Scientific Computing
CS 412
(tutorial class)

Overview

- The vector either as a set of numbers enclosed by parentheses, **u** = (1, 2, 3), or as an M×1 column.
- Vector Arithmetic:

$$a = (ax, ay, az), \quad b = (bx, by, bz)$$

$$a + b = (ax + bx, ay + by, az + bz)$$

$$a - b = (ax - bx, ay - by, az - bz)$$

$$-a = (-ax, -ay, -az), \quad sa = (sax, say, saz)$$

Representing Vectors

- In Maple just write the word vector then specify the content of the vector.

```
> x := Vector([8, 4, 6]);
```

$$x := \begin{bmatrix} 8 \\ 4 \\ 6 \end{bmatrix} \quad (1)$$

- We can get the value in a vector by write the place of value between brackets.

```
> x[1];
```

$$8 \quad (2)$$

3 CS412 Tutorial - Hanouf Aljlayl

Matrix

- Represent the matrices in Maple by writing Matrix, as shown below:

```
> A := Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (3)$$

- To find a specific value in the matrix we can write the number of row and column in brackets.

```
> A[2, 3];
```

$$6 \quad (4)$$

4 CS412 Tutorial - Hanouf Aljlayl

Matrices and Vectors

- Another way to show the vector is simply write a set of numbers between angel brackets.

```
> x := ⟨1, 2, 3⟩;
```

$$x := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad (5)$$

- An alternate, simplified syntax for defining a vector.

```
> B := ⟨⟨1, 2⟩|⟨3, 4⟩⟩;
```

$$B := \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad (6)$$

Vectors into Matrix

- We can define a set of numbers to each vector then combine them into single matrix.

```
> u1 := ⟨1, 3, 4⟩ :
> u2 := ⟨0, 2, 1⟩ :
> u3 := ⟨6, 5, 8⟩ :
> C := ⟨u1|u2|u3⟩;
```

$$C := \begin{bmatrix} 1 & 0 & 6 \\ 3 & 2 & 5 \\ 4 & 1 & 8 \end{bmatrix} \quad (7)$$

Vectors operations

- One of the most common uses of the dot-product is finding the angle between two vectors. The angles are orthogonal if and only if $\mathbf{v} \cdot \mathbf{w} = 0$.

```
> v1 := <1/sqrt(3), 1/sqrt(3), 1/sqrt(3)> :
> v2 := <1/sqrt(2), 0, -1/sqrt(2)> :
> v3 := <1/sqrt(6), -2/sqrt(6), 1/sqrt(6)> :
> v1.v2;
                                  0                              (8)
> v1.v3;
                                  0                              (9)
> v2.v3;
                                  0                              (10)
```


Basics Functions

- Some basics functions in Maple (for e.g. sqrt, sin, cos).

```
> sqrt(2);
                                 √2                              (1)
```

```
> cos(0);
                                  1                              (2)
> sin(0);
                                  0                              (3)
```

Defining Functions

- A function is a "rule" for assigning a value to a number, to define a function in Maple, write *f* then assign the *f* to function by variable.

```
> f := x→x^3;
                        f := x→x^3                    (4)
```

- Some operations in a defined function, we can put an arguments between brackets to compute the function.

```
> f(2.5);
                          15.625                      (5)
```

```
> f(y);
                           y^3                        (6)
```


Compose Functions

- Define another function *g* with variable *y, as shown below:*

```
> g := y→sin(y);
                      g := y→sin(y)                   (7)
```

- Once we have defined two functions we can then compose them.

```
> f(g(x));
                        sin(x)^3                      (8)
> g(f(x));
                        sin(x^3)                      (9)
```

Variables in Functions

- We can define functions of two or more variables, the variables must be in brackets separated by comma.

```
> f := (x, y) → x^2 + y^2;
                    f := (x, y) → x^2 + y^2                    (10)
```

- Compute the function of two variables to get the result of the variables.

```
> f(0, 0);
                              0                                (11)
```

```
> f(3, 4);
                             25                                (12)
```


Sets

- In Maple we can deal with sets, puts the elements of a set between { }. The elements are not in order.

```
> S := {3, 7, 2, 4};
                    S := {2, 3, 4, 7}                          (1)
```

- The repeated element in the set it will left out, as shown below:

```
> T := {1, 2, 2, 3};
                    T := {1, 2, 3}                             (2)
```

- If you want to add an element to a set, use union.

```
> S := S union {9};
                    S := {2, 3, 4, 7, 9}                       (3)
```

Sets

- We can use intersect to get the common element between two sets.

```
> {1, 2, 3, 4} intersect {3, 4, 5, 6};
                          {3, 4}                    (4)
```

- The minus used to delete the common elements between sets.

```
> {1, 2, 3, 4} minus {3, 4, 5, 6};
                          {1, 2}                    (5)
```


Lists

- A list is an ordered set, the elements of a list between [ ].

```
> A := [3, 9, 7, 5, 8];
                    A := [3, 9, 7, 5, 8]            (1)
```

- To get a particular value just write the position of the element in the list.

```
> A[3];
                          7                         (2)
```

```
> A[-2];
                          5                         (3)
```

- We can select specific elements by specify the begin and end position.

```
> A[2..-1];
                       [9, 7, 5, 8]                 (4)
```

Add to list

- Add a value to a list, put the position between [] and assign to a value.

```
> A[2] := 1;
                        A_2 := 1                    (5)
```

- The position [2] now contains the added value, as shown below:

```
> A;
                        [3, 1, 7, 5, 8]             (6)
```


For-loop

- The Maple can deal with loops, in the below for-loop shows a certain values as a list:

```
> for i in [3, 6, 2, 5, 7] do x := i; print('The value of x is', x); end do:
                        The value of x is, 3
                        The value of x is, 6
                        The value of x is, 2
                        The value of x is, 5
                        The value of x is, 7        (1)
```